Sistemas Operativos Distribuidos

Práctica DSM

Objetivo

- Desarrollo de sistema DSM con las siguientes características:
  - Memoria organizada en “objetos” (regiones) compartidos
  - Basado en páginas
  - Modelo de coherencia híbrido: coherencia de entrada
    - Aspecto específico: cada objeto actúa como v. sincro. de sí mismo
    - Proceso que entra en sección crítica de objeto compartido debe ver:
      - Sólo cambios en ese objeto realizados en última SC sobre el mismo
  - Gestor centralizado para mantener info. páginas y control s. crítica
  - Gestor también incluye **todas** las regiones compartidas
  - Política de actualización:
    - *Update* de cliente a gestor
    - *Invalidate* (mediante versiones) en el cliente
- Tecnología de comunicación: sockets
- Llamadas UNIX relevantes: *mmap* y *mprotect*

Arquitectura de servicio e interfaz

- Arquitectura:
  - Aplicaciones enlazan con biblioteca **dsm.c** (debe desarrollarse)
  - Gestor: **gestor.c** y **gestor.h** (deben desarrollarse)
- Interfaz a las aplicaciones:
  - *init_dsm()*: Inicia entorno
  - *void *vincular(nom_obj, tipo_datos, num_elem)*: Crea/asocia región
  - *desvincular(dir)*: Desasocia región
  - *entrar_SC(dir, **modo**)*: Solicitud de entrar s. crítica asociada a región
  - *salir_SC(dir)*: Solicitud de abandonar s. crítica asociada a región

Modo de operación

- Funciones del gestor:
  - Control entrada a sección crítica
  - Mantiene información de regiones
  - Otro cliente más: incluye en su mapa **todas** las regiones compartidas
- Coherencia basada en *write-update* al gestor y versiones:
  - Al salir de sección crítica cliente vuelca a gestor págs. modificadas
    - Se crea nueva versión
  - Clientes y gestor mantienen nº de versión:
    - Al entrar en sec. crítica se invalidan págs. antiguas
  - Algoritmo de versiones a criterio del alumno
    - Enunciado plantea uno basado en nº de versión de región y de págs.
      - Pretende minimizar transferencia de información sobre nº de versión

Protocolo: eventos significativos (1/2)

- Vincular región: Mensaje de cliente C a gestor G
  - G: si existe región, devuelve dirección (nº usuarios++); sino se crea
    - *mmap* con MAP_ANON | MAP_FIXED
    - 1ª región en dir. 0x80000000; siguientes en sucesivas direcciones
  - C: *mmap* similar (pero sin permisos)
- Desvincular región: Mensaje de C a G
  - C: Desproyecta región (*munmap*)
  - G: Si (--nº usuarios==0) región desaparece
- Entrada S.C.: Mensaje de C a G
  - C: Informa a gestor de nº de versión y queda a la espera
  - G: Cuando S.C. libre, desbloquea a C informando de nº de versión
  - C: Quita permiso a págs. viejas, pone sólo lectura a págs. válidas

Protocolo: eventos significativos (2/2)

- Salida S.C.: Mensaje de C a G
  - C: Vuelca páginas modificadas y actualización nº de versión en C
  - G: Copia págs, actualiza versión, da paso a otro C en orden FIFO
- Tratamiento SEGV: Mensaje de C a G; Tratamiento en C:
  - Si dirección incorrecta, error en aplicación.
  - Si página no tenía permisos, pide página a G y pone de sólo lectura
  - Si página tenía permiso R, marca pág. como Mod. y pone RW
  - Recuerde: Si acceso fuera de S.C. resultado impredecible

Parte avanzada

- Entrada a SC en exclusivo (escritor) o compartido (lector):
  - Si solicitud de lector y hay lectores dentro
    - Se otorga acceso si no hay escritores esperando (FIFO)
  - Al salir de S.C.:
    - Si hay varios lectores esperando, acceso a todos previos a un escritor
    - Si primero en espera es escritor, sólo entra ese proceso
  - Detección de errores por escritura de lector: fin de proceso
- Configurable política FIFO o LECTORES_PRIMERO
  - Si solicitud de lector y hay lectores dentro
    - Se otorga acceso aunque haya escritores esperando (LECTORES)
  - Al salir de S.C. un escritor:
    - Si hay varios lectores esperando, acceso a todos
- Si programa "olvida" salir_SC y desvincular
  - Hacerlo automáticamente (uso de función estándar *atexit*)